Load Balance in Cloud Computing using Software Defined Networking

Yassin Abdulkarim Hamdalla Omer
Faculty of Engineering
Al-Neelain University
Khartoum, Sudan
yassincmc50@hotmail.com

Mohamed Ayman Mohammedel-Amin
Faculty-of-Telcommunication
Future University
Khartoum, Sudan
kabantsh@gmail.com

Amin Babiker A. Mustafa
Faculty of Engineering
Al-Neelain University
Khartoum, Sudan
amin31766@gmail.com

2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE) | 978-1-7281-9111-9/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCEEE49695.2021.9429607

Abstract—nowadays due to the high demand for cloud services a load balance mechanism for software defined networking (SDN) in a cloud environment appears to solve the availability and latency issues in the HTTP server. This paper presents a load balance mechanism that has been applied for a software defined networking (SDN) in a cloud environment for the HTTP server which received 6 million requests in total with 400 requests per time. The server will not respond to any further requests that have been transmitted from the client because it is overload. hence the solution is to apply a load balancing mechanism to measure the same parameters after applying load balance. In this scenario, HAproxy has been applied to the cloud environment and all of these 6 million requests with 4 connections per time have been received successfully, the HTTP server's traffic can be distributed among the pool of servers instead of one server this would. Eventually, The HAproxy is giving a good response time with no requests dropped after 6 million requests coming at the same time that help to enhance the availability and improve the latency of HTTP requests by having an HTTP load balancer.

Keywords— **Load balance; Cloud Computing; SDN; Openstack; HAProxy;**

I. INTRODUCTION

The development of new technologies is very useful especially in the field of computer networks. It plays an important role in sharing information across the network that enables people to access their files and resources conveniently. These demands made cloud computing one of the biggest technology trends in the world, which is why the job of researches increases to ensure that ICT infrastructure security with high availability[1,22,23].

Since the initiation of (SDN)[2,14] concepts the old network model has been called "Traditional Network" and the SDN network has been called "Modern Networks". In traditional networks, the control plane and the data plane are doing in the same line and controlled by the networking device switch in Layer 2 and router in Layer3 in the OSI model[3,15].

Traditional networks have two models like peer-to-peer and server clients. The concept of SDN is to divide the control plane from the data plane and have a new device for controlling the traffic which will be called the controller [4,11,12].

A. Software Defined Networking

Software-Defined Networking (SDN) is to enable cloud computing specialists, network engineers, and system administrators to respond quickly to changes in business requirements via a centralized control console which has been called "controller". The only job for the controller is to control network traffic [5,16].

The demand for high availability is encouraging the researchers for having a load balancer to distribute the load among the servers, controllers, or any other resources [6].

There are so many types of controllers depending on which purpose it was developed for and which programming language that it was coded with. are can choose which type, if controller by so many parameters like the language that it is written with or by some other parameter like the throughput, delay, bandwidth, buffer size or the resources consumptions [7]. Through using the network administrator interface, a software defined network infrastructure may be configured without direct modification on physical devices, Accordingly, it can improve the use of network equipment such as load balancing, traffic engineering, and other programmable devices [17,18,19,20,21].

Next-Generation Networks (NGN) was introduced that is managed with a software and it Separates the control plane and data plane from each other by setting it to be managed by a software that is installed in a dedicated computing box called controller[5].

B. LOAD Balance System

Load balancing is a distribution of the network load over the network devices (servers, controllers, routers, and switches), even to handle the whole load to a specific node (or nodes) in case of a failure in one of the devices above. The load balancing mechanism is not only done by specialized hardware or software, but it can also be in every layer of the OSI model [8].

The simplest form of load balancing is DNS load balancing, which is responsible for resolving the IP address into a domain names [4], but load balance here is simple because instead of having only one IP for the server there will be a list of IP (Internet Protocol) in case of failure of one of them, the other will be in charge, but it has a limitation that are cannot identify which server is down. The most important concept that should be considered is servers will be distributed among different geographical areas so that it should be available in case of a natural disaster [4, 9].

The IP addresses will be assigned according to a round robin algorithm. First, we are assigned the server IP address for the first domain name requested. Then the second IP for the second domain name and so on. Until the server's IP round finish (reach to number 'n') then it will start again from the first server and so on as shown in figure (1) [25,26,27].

978-1-7281-9111-9/20/$31.00 ©2020 IEEE

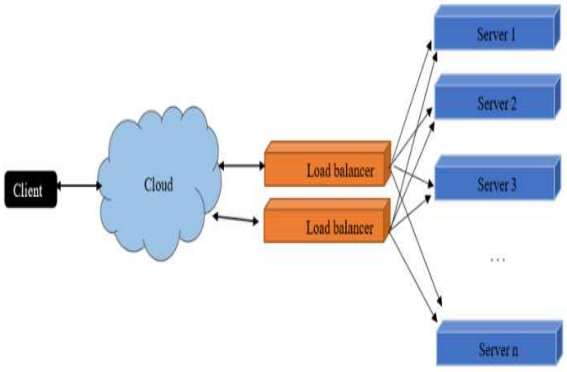

Fig. 1. Load balancing [9]

II. Related Studies

In this section, are review the related literature of the subject of study, and use an Infrastructure topology, and made a software load balancer rather than software. In [9] the authors use round-robin algorithm for enhancing the load balancer as he did write a python script for assigning the load to a distributed pool of servers. Therefore, use a virtual OpenFlow virtual switch (OVS). A load balancer that uses round robin in the DNS to assign one server from the distributed servers statically.

On the other side [6] the authors use a fat-tree topology with eight hosts, ten switches, and 20 links, he built his topology using a python script using Mininet and he chose the load balancer in the application part of SDN (northbound API) which is a software load balancer. This REST API collects the information of the topology and deliver, port statistics as well as traffic instantaneously.

In the ten related work [10] The author uses a basic infrastructure topology with a single switch and two hosts topology and he has made a comparison in the Systematic interpretation of two SDN controllers. He considered the jitter, throughput, and latency to benchmark the differences between two SDN controllers which are Open Networking Operating System (ONOS) and Open Day Light (ODL).

Eventually Ullah, Arif Pointed in [24] depended on a series of scientific papers containing the alphabets of the Load Balance Algorithm Through cloud computing, using virtual machine strategy the performance of cloud computing almost increases.

III. METHODOLOGY

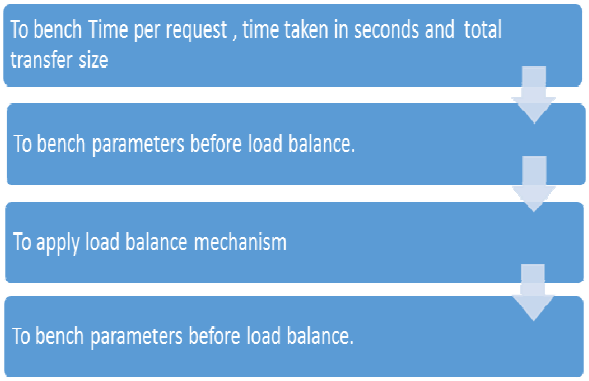

Fig. 2. Implementation of the method steps

This research aims to implement OpenStack cloud and create VMs (Virtual Machines) inside the OpenStack platform, to deploy SDN controller inside one of the VMs to control these VMs. However, to benchmark total transferred size, average requests time, time per requests and taken for tests for ideal case (locally) in the same physical HTTP server, then applying the load balance mechanism

IV. Result and Discussion

In this section of this paper, the results are described as the noteworthy Apache server which shows that can handle more than requests at a time but if they are more than one client using the service then a load balance mechanism should be applied. In table one and figures (1, 2) in these figures if more than 6 million requests are allowing 400 requests per time is coming to the HTTP server the load will be more than what it can handle some of these requests will be dropped as shown in figure 3 and 4 and only 15 requests have been received out of 3 million from client 1 as shown in table (1). In this case, only one client is requesting and there is only one server so that no load balance mechanism is required.

A. One client and one server

Apache server can handle more than requests at a time but if you use only one client but if you use more than one client then a load balance mechanism should be applied in the two figures below (figure 3 and 4). In these figures if more than 6 million requests allowing 400 requests per time is coming to the HTTP server the load will be more than what it can handle some of these requests will be dropped like shown in figure (4) and only 15 requests have been received out of 3 million from client 1 like shown in table (1). In this case, no only one client is requesting and there is only one server so that no load balance mechanism is required.

TABLE I. One Server and One Client

Number of requests	Number of connection	Total transferred size in byte	Time per requests in ms	Time taken for tests in seconds	Transfer rate in Kilobyte per sec
1000000	100	11175000000	40.529	405.288	11106.88
1000000	200	11175000000	80.829	404.145	11102.26
2000000	100	22350000000	40.587	811.733	11105.92
2000000	200	22349977650	80.863	808.634	11106.51
3000000	100	33525000000	40.373	1211.196	11107.39
3000000	200	33525000000	80.966	1214.496	11108.02

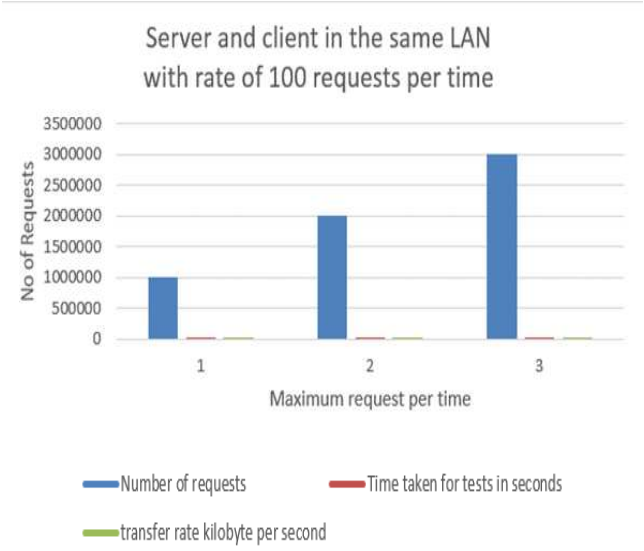

Fig. 3. No LB needed with 100 rate request

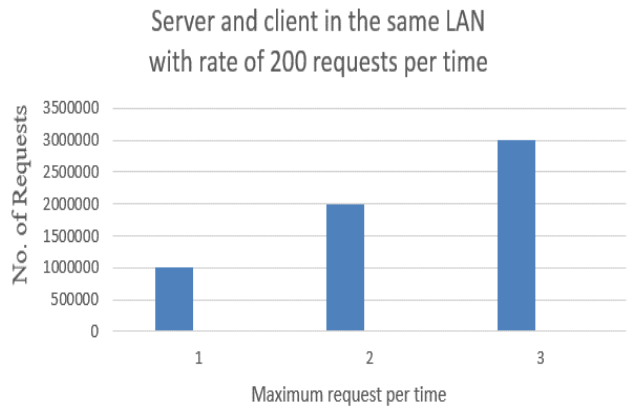

Fig. 4. No LB needed with 200 rate request

All of the 3 million requests have received their reply with the rate of a maximum of 100 requests per time as shown in figure (3) or with a maximum of 200 requests as shown in figure (4).

B. Two clients and one server before load balance

Hence we are using only one client there is no need for load balance because there is only one load. But it is better to use two clients because there will be more than an HTTP session activated in the server. In the table below a benchmark for HTTP requests from client 1 and client 2 at the same time as shown in the table below. Only 384 requests have been received out of 3 million.

TABLE II. CLIENT 1 BEFORE HAPROXY LOAD BALANCE

Number of requests	Number of connection	Total transferred size in byte	Time per requests in ms	Time taken for tests in seconds	Transfer rate in Kilobyte per sec
1000000	100	11192000000	196.527	1965.27	5561.42
1000000	200	11192000000	508.85	1965.22	5561.56
200000	100	22384000	196.469	3929.37	5563.06
0		000		8	
2000000	200	22384000000	392.926	3929.256	5563.23
3000000	100	33576000000	only 384 requests completed	0.488	8594.4
3000000	200	33576000000	no requests came back	0.022	0

When client 1 sends 3 million requests allowing not more than 100 connections per time as shown in figure 5.

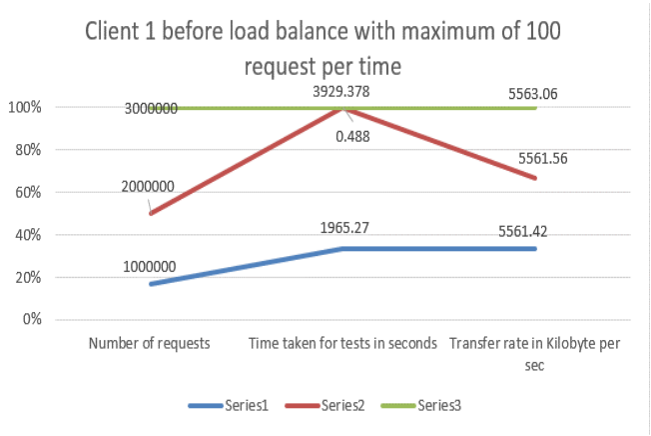

Fig. 5. Client 1 BLB with the rate of 100 request rate

In figure 6 client 1 sends 3 million requests allowing not more than 200 connections per time.

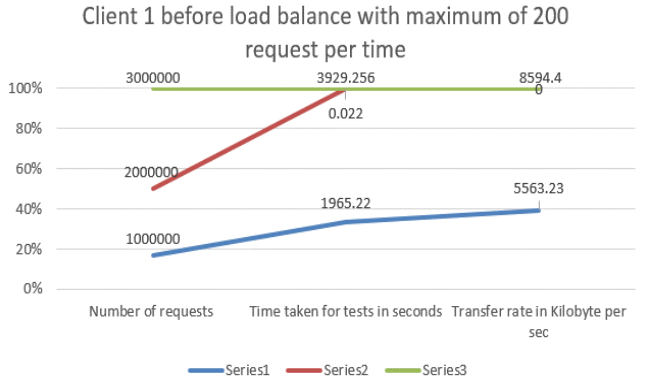

Fig. 6. Client 1 BLB with the rate of 200 request rate

At the same time client 2 is taking the same data with the same client 1 methodology, as shown in table 3.

TABLE III. CLIENT 2 LOAD TEST BEFORE HAPROXY BALANCE

Number of requests	Number of connection	Total transferred size in byte	Time per requests in ms	Time taken for tests in seconds	Transfer rate in Kilobyte per sec
1000000	100	10.43	196.525	1965.25	5561.48
1000000	200	10.43	393.004	1965.021	5562.12

2000000	100	20.85	196.471	3929.428	5562.99
2000000	200	20.85	392.914	3929.14	5563.4
3000000	100	31.28	0	0.023	7519.47
3000000	200	31.28	0	0.022	0

As a conclusion for the above tables, HTTP servers cannot handle more than 6 million (3 million for the 1st client and 3 for the second) requests per time so that a load balance mechanism should be applied. are use HAproxy for high availability of the HTTP servers (Apache2) and a round robin algorithm has been applied to a pool of servers

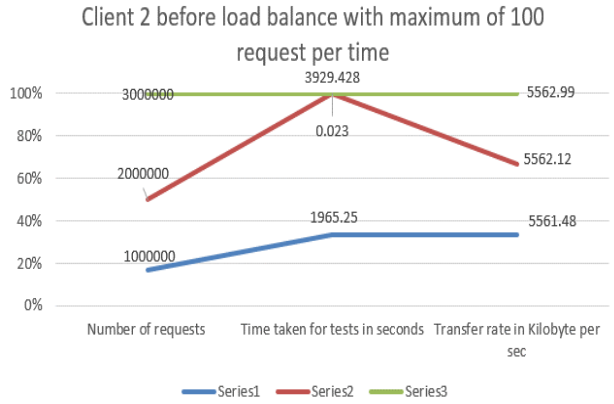

Fig. 7. Client 2 BLB with the rate of 100 request rate

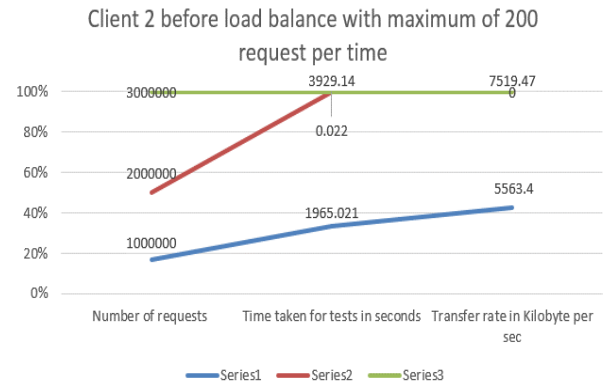

Fig. 8. Client 2 BLB with the rate of 200 request rate

At the same time while client 1 is sending its 3 million requests client 2 is also sending 3 million requests simultaneously. When the HTTP server reached its limit (6 million requests which are 400 requests per time) it starts to misbehave and it did a response to only 15 requests out of 3 million as shown in table (3).

When the HTTP server has received 6 million requests in total with 400 requests per time. The server will not respond to any further requests that have been transmitted from the client because it is overloaded.

As mentioned earlier HTTP server cannot handle more than 6 million requests with a maximum of 400 requests per time simultaneously same like client 1, client 2 has requested the server at the same time but when it reaches 2 million requests out of 3 million most of HTTP requests has been dropped like shown in figure 10 in the bottom of this page, when allowing 100 requests per time.

C. Two clients and one server after load balance

After applying the HAproxy load balance mechanism the results of client 1 are given in table 4 and client 2 response is given in table 4.

TABLE IV. CLIENT 1 LOAD TEST AFTER APPLYING LOAD BALANCE

Number of requests	Number of connection	Total transferred size in byte	Time per requests in ms	Time taken for tests in seconds	Transfer rate in Kilobyte per sec
1000000	100	387000000	13.483	134.833	2802.94
1000000	200	387000000	26.843	134.215	2815.86
2000000	100	774000000	13.476	269.528	2804.38
2000000	200	774000000	26.933	269.332	2806.42
3000000	100	1161000000	13.503	405.094	2798.83
3000000	200	1161000000	27.002	405.034	2799.24

As seen in table 4 the client 1 and 2 also receive 1 million requests with a maximum of 100 then 200 another as shown in figures 9 and 10 as seen from the figure the servers keep responding from the pool of servers with no dropped connection like before applying the load balance.

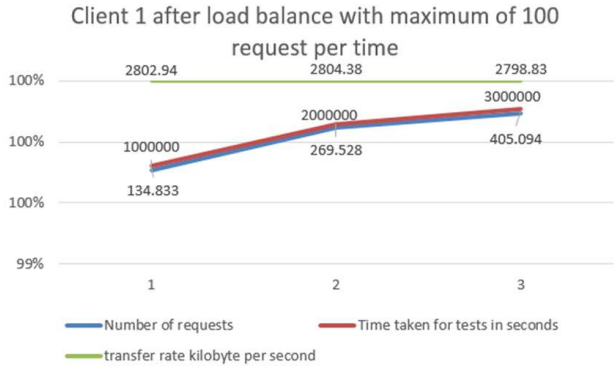

Fig. 9. Client 1 ALB with 100 request rate

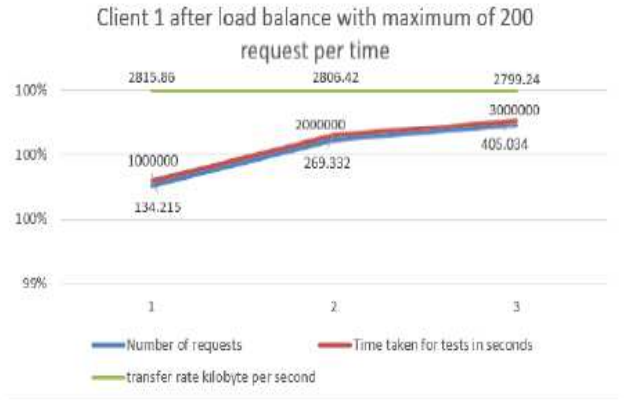

Fig. 10. Client 1 ALB with 200 request rate

Meanwhile, client 2 has followed the same methodology as client 1 as shown in table 5

TABLE V. CLIENT 2 LOAD TEST AFTER HAPROXY LOAD BALANCE

Number of requests	Number of connection	Total transferred size in byte	Time per requests in ms	Time taken for tests in seconds	Transfer rate in Kilobyte per sec
1000000	100	387000000	13.483	134.143	2817.36
1000000	200	387000000	26.837	134.215	2816.49
2000000	100	774000000	13.485	269.691	2802.69
2000000	200	774000000	26.94	269.405	2805.67
3000000	100	1161000000	13.504	405.116	2798.68
3000000	200	1161000000	27.004	405.058	2799.07

This table (5) has been transferred into two figure 10 and 11.

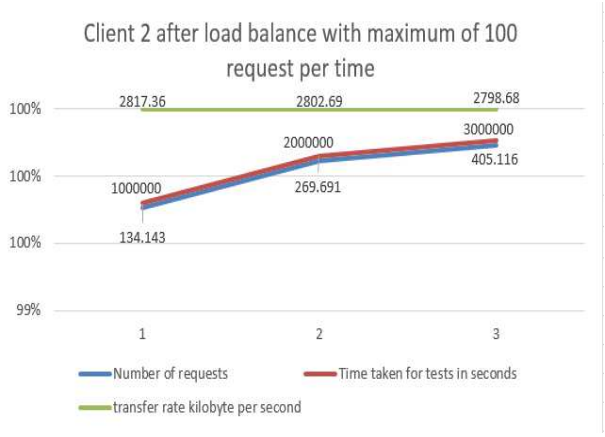

Fig. 11. Client 2 ALB with 100 request rate

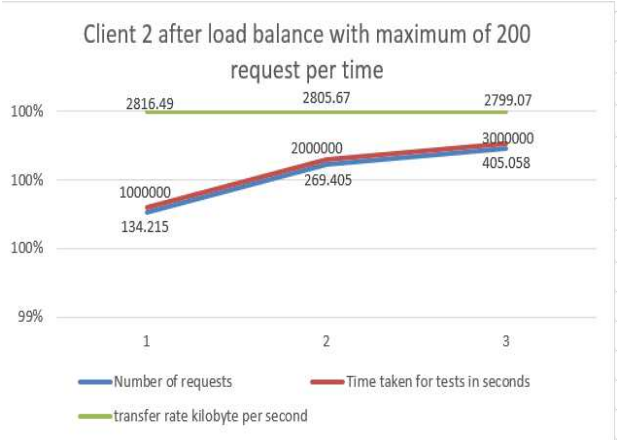

Fig. 12. Client 2 ALB with 200 request rate

V. CONCLUSION

From the result, HAproxy is giving a good response time with no requests dropped after 6 million requests coming at the same time for requests, in this scenario it is giving a very good result for both client 1 and client 2 on using Round Robin algorithm, this round robin algorithm can be used each pool is having the same resources and it is identical, but in case of using non-identical pools it is better to use weighted round robin and give the higher resources specification a higher weight, less resource a less weight in the algorithm.

Nevertheless, it is important to highlight that the HTTP server's traffic can be distributed among the pool of servers instead of one server. Moreover, after applying the load balancer technique in the HTTP server it was found that the availability has been increased as well as the latency. Eventually, the OpenStack cloud has so many difficulties with the installation and configuration. It is better to know the OpenStack cloud topology or whether are wants to install in on a single node or to a distributed system that each machine is responsible for only one module or more and to choose carefully which Linux distribution will be installed on.

VI. RECOMMENDATIONS

Recommended that if the enterprise wants to provide any Load Balance on layer 4 or layer 3 they may need to use another proxy, depending on the topology and the mechanism of load balance.

Consider the type of service whether it is real-time data or not, or whether the reliability of the data is a big aspect or not. So if load balance is needed on the service itself it is better to use an application proxy. In this paper, the focus of load balance is only on an HTTP protocol that is built on top of TCP but for further researchers, this method can be applied in more than a protocol like ICMP and UDP. In this chapter the methodology of benchmarking the HTTP request of HAproxy, also It is not recommended to let the OpenStack cloud take a dynamic IP, therefore it is better to let give the cloud nodes and static IP or let it take only one IP from the DHCP server with an infinite lease time.

REFERENCES

[1] Armbrust, M., et al., A view of cloud computing. Communications of the ACM, 2010. 53(4): p. 50-58.

[2] McKeown, N., Software-defined networking. INFOCOM keynote talk, 2009. 17(2): p. 30-32.

[3] Wetteroth, D., OSI reference model for telecommunications. Vol. 396. 2002: McGraw-Hill New York.

[4] Ranjani, S., Dynamic Load Balancing using Software Defined Networks.

[5] Sezer, S., et al., Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine, 2013. 51(7): p. 36-43.

[6] Zakia, U. and H.B. Yedder. Dynamic load balancing in SDN-based data center networks. in Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2017 8th IEEE Annual. 2017. IEEE.

[7] Subramanian, S. and S. Voruganti, Software-Defined Networking (SDN) with OpenStack, ed. C.P. Publishing. 2016, Livery Place: Packt Publishing Ltd. 35 Livery Street. 189.

[8] Briscoe, N., Understanding the OSI 7-layer model. PC Network Advisor, 2000. 120(2).

[9] Senthil Ganesh, N. and S. Ranjani, Dynamic load balancing using software defined networks. International Journal of Computer Applications (0975-8887), 2015.

[10] Yousif., Y.A., SDN Controller Performance analysis in Datacenter., in Future university journal. 2018.

[11] p. kumari and D. Thakur, "Load Balancing in Software Defined Network.," 2017.

[12] Themba Shozi, Pragasen Mudali, Matthew O. Adigun, Hlabishi Kobo. "A Data Plane Multi-Path Load Balancing Mechanism for Hybrid Software Defined Networks in Different Topologies" , 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), 2019

[13] Zheng Zhao, Fenlin Liu, Daofu Gong, Lin Chen, Fei Xiang, Yan Li. "An SDN-based IP hopping communication scheme against scanning attack" , 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017

[14] Fei Xiang, Yan Li. "An SDN-based IP hopping communication scheme against scanning attack" , 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017

[15] D. Francis Xavier Christopher, C. Divya. "Chapter 30 Address Resolution Protocol Based Attacks: Prevention and Detection Schemes" , Springer Science and Business Media LLC, 2020

[16] Nadeau, Thomas D., and Ken Gray. SDN: Software Defined Networks: an authoritative review of network programmability technologies. " O'Reilly Media, Inc.", 2013.

[17] Calin, Doru, and Hyunwoo Nam. "Optimizing quality of service in a content distribution network using software defined networking." U.S. Patent No. 10,028,167. 17 Jul. 2018.

[18] [2] H. T. Zaw, A. H. Maw, "Traffic management with elephant flow detection in software defined networks (SDN)," International Journal of Electrical and Computer Engineering, vol. 9, no. 4, pp. 3203-3211, August 2019. doi: 10.11591/ijece.v9i4.pp3203-3211.

[19] Madhusanka Liyanage; Andrei Gurtov; Mika Ylianttila, "Software Defined Networking Concepts," in Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture, John Wiley & Sons, Ltd., pp. 21-44, 2015.

[20] Jain R., Paul S. "Network Virtualization and Software Defined Networking for Cloud Computing: A Survey," IEEE Communications Magazine, vol. 51, no. 11, pp. 24-31, November 2013.

[21] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan 2015.

[22] Ullah, Arif. "Artificial bee colony algorithm used for load balancing in cloud computing." IAES International Journal of Artificial Intelligence 8.2 (2019): 156.

[23] Umar, S., & Baseer, S. (2016, August). Perception of cloud computing in universities of Peshawar, Pakistan. In Innovative Computing Technology (INTECH), 2016 Sixth International Conference on (pp. 87-91). IEEE.

[24] Ullah, A., Nawi, N. M., Shahzad, A., Khan, S. N., & Aamir, M. (2017). An E-learning System in Malaysia based on Green Computing and Energy Level. JOIV: International Journal on Informatics Visualization, 1(4-2), 184-187.

[25] Devi, D. Chitra, and V. Rhymend Uthariaraj. "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks." The scientific world journal 2016 (2016).

[26] Pradhan, Pandaba, Prafulla Ku Behera, and B. N. B. Ray. "Modified round robin algorithm for resource allocation in cloud computing." Procedia Computer Science 85 (2016): 878-890.

[27] Mahajan, Komal, Ansuyia Makroo, and Deepak Dahiya. "Round robin with server affinity: a VM load balancing algorithm for cloud based infrastructure." (2013).